HANDPRINTED CHARACTER RECOGNITION BY VORONOI SKELETONS

BY

NIRANJAN R. MAYYA

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1994

To my father, B. Ramakrishna Mayya, my mother, B. Nalini Mayya, and my sisters, Usha, Chitkala and Giribala Mayya.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my advisor and chairman of my supervisory committee, Dr. Andrew F. Laine for his guidance, support and encouragement for the last one year. His confidence in me has been unflagging, and I deeply appreciate that.

I was fortunate to interact with Dr. V. T. Rajan of the IBM T. J. Watson Research Center, who was my research mentor during my summer internship there. I thank him for his advice, support and the many hours of fruitful discussion we have had. I thank my cochairman, Dr. Baba C. Vemuri, and the other members of my supervisory committee, Dr. Gerhard Ritter, Dr. Sartaj Sahni, Dr. Murali Rao and Dr. Richard Kiltie, for their service on my committee.

Thanks are due Sergio Schuler and Jian Fan, my fellow students in the department, for their advice and comments while I was working on my dissertation.

My friends at the University of Florida have been my "family" in Gainesville, and I thank them for their encouragement and moral support. In particular, I would like to mention my roommate, Ravikanth Malladi, whose friendship and loyalty have been been particularly heartwarming. To my other friends Chitrank Seshadri, Vidhya Gangar, Roxana Dastur and Pankaj Aggarwal, I would like to express my thanks and appreciation for all the good times we have shared.

Prem Pahlajrai has been a source of support from afar and kept my spirits up when the going seemed bleak. I thank him for being a good friend.

Finally, I would like to express my deep thanks and gratitude to my family, whose love and support have stood me in good stead during my many years in Gainesville.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

HANDPRINTED CHARACTER RECOGNITION BY VORONOI SKELETONS

By

Niranjan R. Mayya

August 1994

Chairman: Andrew F. Laine
Major Department: Computer and Information Sciences

This dissertation makes contributions to two key areas in computer vision, namely the areas of shape representation and pattern classification. In the first part of this work, we describe a new shape representation framework based on what are termed Voronoi skeletons. The skeleton of a planar shape is the reduction of a shape into a one-dimensional representation. Skeletons have been widely used in shape modeling as they serve as compact descriptors of the "natural" shape of an object, which well describes its global topological and geometric properties. Skeletonization techniques abound in the literature, with each technique having its own strengths and weaknesses. In this dissertation we present a new skeletonization algorithm that improves on past methods by overcoming several limitations in existing techniques. Our method draws on concepts from computational geometry; in particular we employ a geometric construct called the Voronoi diagram to compute skeletons of polygonal shapes accurately. Our approach employs a novel algorithm for computing the Voronoi diagram of a polygon with holes. We show that Voronoi skeletons provide shape descriptors superior to those that use skeletons computed from discrete implementations because they are accurate, preserve connectivity and are characterized by the Euclidean metric.

The second part of the dissertation focusses on the problem of isolated hand-printed character recognition. Character recognition is an area that has applications in document understanding, in automated postal sorting and more recently in pen-based computing. Research on this topic continues at a hectic pace, largely because the error rates currently achieved in recognition are not up to commercially acceptable standards. Here we present a technique to classify character patterns using a neural network trained on the Voronoi skeleton representations described above. Compared to traditional skeletonization techniques, we suggest that shape representations based on Voronoi skeletons may increase the reliability of production quality character recognition systems. Results on two different data sets are presented and indicate the efficacy of our character recognition system.

CHAPTER 1
INTRODUCTION

Image analysis in computer vision is a process of discovering, identifying, and understanding patterns that are relevant to the performance of an image-based task. One of the principal goals of computerized image analysis [17] is to program the computer with the ability to simulate or approximate similar capabilities in humans. The techniques involved in most image analysis systems can be broadly classified into three sub areas. These subdivisions have no definitive boundaries. We discuss here a taxonomy as described in [17]. These are i) low-level processing, ii) intermediate-level processing and iii) high-level processing. Low-level functions include preprocessing techniques. Image acquisition may also be considered a low-level function. The input is in general a two-dimensional image represented as an intensity map. The output of this stage is a processed image where the processing techniques typically include noise reduction or image deblurring.

Intermediate-level processing deals with the task of extracting and characterizing components in an image. These techniques include segmentation and description. Segmentation subdivides an image into its constituent parts or objects, using a variety of techniques, based either on discontinuity or similarity properties of gray-level values in an image. Techniques for description take an image segmented into regions, and represent and describe them in a form suitable for higher-level processing. We will describe techniques for representation in further detail in the next chapter.

Recognition and interpretation of images form the final step in image analysis systems. The success or failure of this step is determined by subjective criteria in general. This implies that the techniques employed in recognition and interpretation (particularly interpretation) are less precise than those employed in the earlier two stages, and the techniques do not always have a well-defined theoretic formulation. An important goal in this stage is an effective organization and use of knowledge about a problem domain. Recognition methods are of two types: a) decision theoretic methods for recognition and b) structural methods for recognition. Decision theoretic methods are based on representing patterns in vector form and seeking approaches that group and classify patterns into various classes. The approaches used include minimum distance classifiers, Bayes classifiers and neural networks. In structural recognition, patterns are represented in symbolic form and recognition techniques based on symbol matching or on models that treat symbol patterns as sentences from an artificial language.

This dissertation makes contributions to two of these key areas in computer vision, namely the areas of shape representation and pattern classification. In the first part of this work, we describe a new shape representation framework based on what are termed Voronoi skeletons [9, 38]. The skeleton of a planar shape (also known as the symmetric axis or the medial axis transform) is the reduction of a shape into a one-dimensional representation. (Formal definitions follow in Chapter 2.) Skeletons have been widely used in shape modeling [44, 42, 8, 6, 10, 14, 37] as they serve as compact descriptors of the "natural" shape of an object, which well describes its global topological and geometric properties. In particular for polygonal shapes, each skeletal edge is the locus of equidistance of two boundary edges. For planar shapes in general, each skeletal point is equidistant and closest to two or more points on the shapes bounding contour. Skeletonization techniques abound in the literature, with each technique having its own strengths and weaknesses. In this dissertation we present a new skeletonization algorithm that improves on past methods by overcoming several limitations in existing techniques. Our method draws on concepts from computational geometry; in particular we employ a geometric construct called the Voronoi diagram to compute skeletons of polygonal shapes accurately. Our approach employs a novel algorithm for computing the Voronoi diagram of a polygon with holes. We show that Voronoi skeletons provide shape descriptors superior to those that use skeletons computed from discrete implementations because they are accurate (with respect to the continuous definition of the skeleton), preserve connectivity and are characterized by the Euclidean metric.

The second part of the dissertation focusses on the problem of isolated hand-printed character recognition. Character recognition is an area that has applications in document understanding, in automated postal sorting and, recently, in pen-based computing. Research on this topic continues at a hectic pace, largely because the error rates currently achieved in recognition are not up to commercially acceptable standards. Here we present a technique to classify character patterns using a neural network approach. Our neural network is trained on the Voronoi skeleton representations described above. Results on two different data sets are presented and indicate the efficacy of our character recognition system.

The rest of the material in this dissertation is as follows. In chapter 2, we introduce the shape representation framework and present related work in the field. Chapter 3 presents the details of the algorithm to compute the Voronoi diagram of a polygon, followed by the algorithm for skeletonization in Chapter 4. In Chapter 5, we discuss the character recognition problem in some detail and present our approach and our classification results in Chapter 6. We conclude in Chapter 7 with a discussion of future research directions.

CHAPTER 2
SHAPE REPRESENTATION AND DESCRIPTION

The representation and description of planar shapes or regions that have been segmented out of an image are early steps in the operation of most computer vision systems; they serve as a precursor to higher level tasks such as recognition and/or interpretation.

Numerous approaches to represent and describe planar shapes abound in the literature and have been applied to diverse problems in computer vision. In general, techniques are either region-based or boundary-based methods. However, a representation scheme is only the first step; it must be followed by a description of the image that utilizes this representation. For example, in a boundary representation of a region, the boundary may be described by features such as its length or the number of concavities.

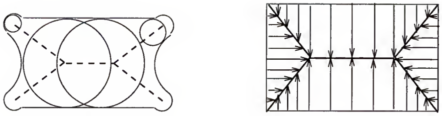
Boundary-based methods analyze objects by representing the bounding contour in a variety of ways; traditional methods include chain-codes, polygonal approximations, signatures and convex hulls. Boundary descriptors use this information to describe an object in terms of critical features of the bounding contour, such as its length, and points of curvature and slope discontinuity.

Region-based methods represent objects in terms of their internal characteristics (the pixels composing a region). Simple region descriptors include the area of the region, the mean and median gray levels, and the minimum and maximum gray-level values. The area provides a measure of compactness, which is defined as $(perimeter^2/area)$. Topological descriptors are useful for global descriptions of regions in the image plane. Topological properties are those that are invariant under any deformation, except those that tear or join the figure. Descriptors often used include the number of holes in a given planar shape (given by the Euler number of a shape), the number of connected components etc.

The symmetric or medial axis transformation (SAT, MAT), also known as the skeleton is another important member of the class of region descriptors. The skeleton is a a one dimensional representation of a planar shape that includes the axes of symmetry of the shape. The SAT has several features, the most important of which is that it is a compact descriptor of the shape of an object. In the remainder of this section, we look at skeletons and skeletonization algorithms in greater detail.

The Symmetric Axis Transform was first introduced by Blum [5] using a circular primitive. In this definition, the skeleton of a shape is consists of the locus of centers of maximally inscribed discs and the radius at each point known as the radius function.

There are several equivalent definitions for the SAT; the most well known of which is the prairie fire analogy [5]. If one applies fire to all the sides of P and lets the fire propagate at constant speed, then the skeleton is the locus of points where fire wave fronts meet. The time taken for the wavefronts to meet, assuming unit velocity provides the radius function. See Figure 2.1.


(a) Locus of maximal discs

(b) Prairie fire analogy

Figure 2.1. Symmetric Axis Transform

An equivalent definition of skeletons is provided by Pavlidis [39]: "Let R be a plane set, B its boundary, and P a point in R. A nearest neighbor of P on B is a point M in B such that there is no other point in B whose distance from P is less than the distance PM. If P has more than one nearest neighbor, then P is said to be a *skeletal point* of R. The union of all skeletal points is called the skeleton or medial axis of R."

The skeleton of a planar object relates the internal structure of the object to significant boundary features. For polygonal shapes, each skeletal edge is the locus of equidistance between two boundary elements. Leyton [29] has shown, by his theorem on symmetry-curvature duality, that a stronger result holds for the more general case of regular curves. His theorem states: "Any segment of a smooth planar curve, bounded by two consecutive curvature extrema of the same type, has a unique symmetry axis, and the axis terminates at the curvature extremum of the opposite type." Thus in a sense, the SAT serves as a bridge between boundary-based and region-based approaches to shape representation. It is a compact descriptor of the "natural" shape of an object, which well describes its global topological and geometric properties. Skeletons have been used in higher level computer vision tasks such as object/character recognition, as they provide a more explicit and compact representation compared to the original intensity map of an image. Finally, the reduction from a two-dimensional intensity map to a one-dimensional line drawing allows for applications of data compression, where compressed images need to be transmitted to remote locations.

Skeletons have been used to solve problems in object recognition [38], to model and characterize the geometry of nonrigid biological forms [14, 28] and in character recognition [52, 23]. In the next section we discuss algorithms for skeletonization and the problems associated with each of these techniques.

2.1 Algorithms for Skeletonization

Recall that a point p belongs to the skeleton of a shape R, if p has two or more "closest" neighbors on the boundary B of R. The concept of "closest" neighbor is closely associated with the metric of distance under consideration, and the resulting skeletons are influenced by the choice of the distance metric. The Euclidean metric is one that correctly characterizes the continuous definition of skeletons provided by Blum. However, preserving Euclidean metrics in a discrete world where shapes are embedded on a two-dimensional grid has been surprisingly difficult to achieve.

Most implementations for computing the skeleton to date use discrete space concepts that only approximate Blum's definition. Another important property for skeletons is to achieve connectivity. While this seems a trivial property from the definition of the skeleton, the discrete nature of images ensures that connectivity is not guaranteed. The various algorithms and implementations that have been published to date can be classified into three different groups [28, 38]:

- Topological thinning.
- Medial Axis extraction from a distance map.
- Analytic computation of a skeleton based on an approximation of the bounding contour of the planar shape.

2.1.1 Thinning Algorithms

A large class of thinning algorithms examines the topological relevance of object pixels rather than the metric properties of the shape. Some examples of thinning algorithms are given in [18, 40, 1, 14], while a survey of thinning methodologies is provided by Lam et al. [24]. In this technique, object pixels are repetitively tested and subsequently deleted, whenever their removal does not alter the topology of the thinned shape. The advantage of this approach to skeletonization is that they ensure connected skeletons using fast algorithms. However the prime disadvantage is that the discrete domain gives rise to non-Euclidean metrics. Different thinning algorithms applied on the same image can result in skeletons that vary. These problems are in part due to different pixel "removal conditions" that are defined in terms of local configurations.

2.1.2 Medial Axis Extraction from a Distance Map

The second method of skeleton extraction requires the computation of a distance map–that is, to determine, for each point inside the object, the distance of the closest point from its boundary. Representative algorithms for this method of skeletonization may be found in [2, 7, 15, 33]. Depending on the metric used for distance, a wide array of possible distance maps can be obtained. Unfortunately, the easiest and simplest algorithms are those based on non-Euclidean metrics, which lead to skeletons that are not very accurate in terms of the fire front paradigm. However, algorithms to compute correct Euclidean distance maps exist [12]. Skeletonization algorithms using quasi-Euclidean and Euclidean maps follow ridges in the distance map to obtain the skeleton. The problem with this method is that connectivity is not guaranteed [28].

2.1.3 Analytic Computation of the Symmetric Axes

The third method involves computation of the symmetric axes by a direct analytical method based on the polygonal approximation of a shape. Early work using this approach was by Montanari [34], who solves systems of linear equations to compute loci of equidistant points.

The Voronoi diagram is a useful geometric structure that contains the entire planar proximity information of a set of points [3]. This allows for easy computation of the distance map from the Voronoi diagram. Further, the Voronoi diagram of the boundary line segments of a polygon is closely associated to its medial axis. In fact, the medial axis is exactly contained in the set of Voronoi edges of the polygon, and is obtained by deleting the two Voronoi edges incident with each concave vertex

[26]. Thus construction of the Voronoi diagram is one technique for skeletonization of polygonal shapes.

2.2 Skeletons Derived from Voronoi Diagrams

Using the Voronoi diagram to compute the skeleton of a polygonal shape is attractive because it results in skeletons that are connected while retaining Euclidean metrics. Furthermore, we obtain a medial axis that is exact upto the polygonal approximation of the shape. Thus we may reconstruct exactly an original polygon from its skeleton, (invertibility, one-to-one mapping). Finally, algorithms to compute the Voronoi diagram (and hence the skeleton) are faster than approaches that compute a distance map. (Optimal algorithms to compute the Voronoi diagram take $O(B\log(B))$ time, where B is the number of segments on the polygon. Algorithms to compute the distance map take $O(N^2)$ time for a NxN image.

However, we observe that there are some disadvantages of using Voronoi diagrams to derive skeletons. We suggest that any method that utilizes Voronoi diagrams of polygons to compute skeletons must overcome the disadvantages listed below before it can be of practical value.

1. Natural shapes are nonpolygonal. Thus, accurate polygonal approximations of such shapes are required in order to compute skeletons without loss of accuracy.

2. The skeleton of a many sided polygon (of very short sides) will have a large number of redundant edges because of the Voronoi edges at these vertices. This results in an increase in the complexity of the skeleton, without significant addition of any shape information.

3. Finally, robust and practical algorithms (those affording ease of implementation) for Voronoi diagram construction of polygons are not very common. Most existing algorithms make assumptions about cocircularity of no more than three points and colinearity of no more than two. Practical algorithms should not make these assumptions since these constraints (colinearity or cocircularity) are difficult to satisfy in practice.

In the next section we present an outline of our approach that overcomes the above disadvantages.

2.3 Voronoi Skeletons for Shape Representation

In this dissertation we present a shape representation framework based on Voronoi skeletons. The basis of our approach is a new algorithm for computing the Voronoi diagram of a polygon. Our skeletonization algorithm retains the advantages of Voronoi diagrams described in the previous section, (connectivity, Euclidean metrics and accuracy). This approach is thus a marked improvement over traditional skeletonization methods. Furthermore we avoid the pitfalls of Voronoi skeletons by overcoming the disadvantages identified in the previous section.

The original contributions made here may be enumerated as follows:

1. A new algorithm for the construction of the Voronoi diagram of a polygon with holes. This algorithm is practical and robust. Our Voronoi diagram algorithm is simple to implement. Assumptions about points being in general position (no three points being colinear or no four points being cocircular) are unnecessary. Unlike most geometric algorithms, special cases such as cocircularity of greater

than three points or colinearity of more than two points are handled elegantly. In addition, our algorithm features a robust numerical scheme to compute non-linear parabolic edges that avoids having to solve equations of degree greater than two.

2. An accurate scheme for skeletonization that retains the advantages of Voronoi skeletons while overcoming its unattractive features. In particular, the scheme provides skeletons that maintain Euclidean metrics without sacrificing connectivity. The method is relatively stable with respect to being invariant to perturbations along the boundary. This is achieved by pruning redundant or spurious edges that do not significantly add to shape information. We will see that the pruning step is a simple consequence of the Voronoi diagram algorithm and does not require any postprocessing. Further, our approach handles shapes with and without holes in a seamless manner, without resorting to special initialization.

In the next chapter we describe the Voronoi diagram algorithm and then proceed to discuss our skeletonization scheme.

CHAPTER 3
THE VORONOI DIAGRAM OF A POLYGON

The Voronoi diagram of a set of sites in two dimensions is the partition of the plane into regions, each region i containing the set of points in the plane closest to the site i.

In the most common case, which has been exhaustively researched in the past decade [43], the sites under consideration are points in the plane. In this case, edges of the diagram are straight-line segments that are perpendicular bisectors of pairs of sites. Optimal algorithms as well as robust implementations have been devised for this version of the problem.

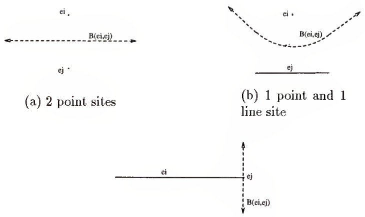
The notion of a site has been generalized to include a collection of two-dimensional objects such as line segments and circular arcs [53]. In the case of line segments, the edges of the ensuing Voronoi diagram are not just straight-line segments, but also arcs of parabola, since they separate the loci of proximity of object pairs of the types $(point, point)$, $(line, line)$, $(point, line)$; the last of these pairs gives rise to arcs of parabola. In the case of circular arcs, the Voronoi bisectors can be general conic sections. Much research effort has also been expended in this direction. In particular, Lee and Drysdale [27] have a $O(n \log^2 n)$ time algorithm for the Voronoi diagram of a mixed collection of n objects (including line segments and circles). Kirkpatrick presented the first optimal $\Theta(n \log n)$ solution; Fortune [16] and Yap [53] also have optimal algorithms. Fortune's algorithm computes Voronoi diagrams of line segments; Yap considers simple curve segments. While some of these algorithms are optimal in terms of time complexity, they are not amenable to simple implementation. Some work has been performed in this direction as well, notably the work of Srinivasan and Nackman [48] and Meshkat and Sakkas [31], where the authors present a simple algorithm for the Voronoi diagram of multiply connected polygonal domains (polygons with holes).

In this section we present an algorithm that solves a slightly more general version of the problem than that of Srinivasan and Nackman. While the working examples shown are those of polygons with holes, the algorithm presented here can compute the Voronoi diagram of a planar straight line graph. The algorithm is easy to implement and has been fully implemented and tested. Another important feature is its robustness; the algorithm handles the standard degenerate cases inherent in most computational geometry algorithms (more than two colinear or more than three cocircular points). This facilitates its use in most practical applications.

3.1 Preliminaries

In this section, we will introduce some definitions and notation. Definitions 1 through 8 pertain to the definition of polygonal domains and Voronoi diagrams and are taken from Srinivasan and Nackman [48].

Definition 1 *A closed line segment $[a, b]$ is the union of two endpoints a and b and the open line segment (a, b).*



(a) 2 point sites

(b) 1 point and 1 line site

(c) the point site is an end point of the line site

Figure 3.1. Bisectors of e_i and e_j

Definition 2 *A multiply connected polygonal domain P is the closure of a nonempty, bounded, connected, open (in the relative topology) subset of $\mathcal{R}^2$ whose boundary is the union of a finite number of closed line segments.*

The boundary of P, denoted by δP, consists of one or more disjoint subsets. The outer boundary of the polygonal domain is denoted by δP_0 and contains P. The inner boundaries represent the holes of the polygonal region and are denoted by δP_i, $1 \leq i \leq H$, where H denotes the number of holes.

Definition 3 *The vertices of δP are the points of intersection of the closed line segments that constitute δP. The edges of δP are the open line segments obtained by deleting the endpoints of the closed line segments that constitute δP*

Definition 4 *Given a simple polygon, $G = (q_0, q_1, ...q_{n-1})$, a vertex q_i is called convex if the internal angle at q_i is less than π and is called concave otherwise.*

<u>*Definition 5*</u> *The projection $p(q,[a,b])$ of a point q onto a closed segment $[a,b]$ is the intersection of the line through a and b and the line perpendicular to $[a,b]$ and passing through q.*

<u>*Definition 6*</u> *The bisector $B(e_i,e_j)$ of two sites e_i and e_j is the locus of points equidistant from e_i and e_j.*

<u>*Definition 7*</u> *The half-plane $h(e_i,e_j)$ is the set of points closer to site e_i than to site e_j. Its complement, $\bar{h}(e_i,e_j)$, is the set of points not closer to site e_i than to e_j.*

The nature of the bisector is determined by the nature of the sites (point, line). In particular, when both e_i and e_j are point sites, the bisector is a straight line (the perpendicular bisector of e_i and e_j. When one of the sites is a point and the other is an open line segment, the bisector is in general a parabolic arc. For the special case where the point is one of the endpoints of the open line segment, the bisector is a straight line passing through the point and perpendicular to the open line segment. See Figure 3.1.

<u>*Definition 8*</u> *Given a set of sites S and a site e_i, $e_i \notin S$, the Voronoi region of e_i with respect to S, denoted by $V(e_i,S)$, is the set of all points closer to e_i than to any site in S.*

<u>*Lemma 1*</u> $V(e_i,S) = \cap_{e_j \in S} h(e_i,e_j)$

Proof: Corollary 2, Lemma 1 of [48] ■

<u>*Definition 9*</u> *The Voronoi diagram, VOD(S), of a set of elements $S = e_i$ is given by* $\cup_{e_i \in S} V(e_i, S - e_i)$

3.2 Primitives used in the Algorithm

A site e_i can be either a point or an open line segment. Each such site is associated with the following information.

1. A contact point x_{p_i}. For a point site, the contact point is the site itself; for a line site, the contact point is one of the endpoints.

2. A distance function $d_i(x_0)$, that returns the square of the distance of a point x_0 from site e_i. For a line site, the distance of a point to the site is defined as the distance from the point x_0 to its projection on the line site e_i. Given this definition, we can define the distance of a point to any site (either a point or a line) as follows, Given a line site, let $\vec{n_l}$ be its unit vector. If I is the identity matrix, and T denotes the transpose of a matrix, we define a 2×2 matrix M as

$$\begin{aligned} M &= I & &\text{for a point site,} \\ &= I - \vec{n_l}\,\vec{n_l}^T & &\text{for a line site} \end{aligned}$$

Now we define the distance function as:

$$d_i(x_0) = (x_0 - x_{p_i})^T\, M\, (x_0 - x_{p_i})$$

3. A gradient vector $g(x_0)$, which is defined by:

$$g_i(x_0) = 2(x_0 - x_{p_i})^T \, M$$

4. The quadratic polynomial $f_i(x_0 + \vec{v}\tau)$, that gives the square of the distance of the point $(x_0 + \vec{v}\tau)$ from the site i. Here $\vec{v}$ is a direction vector, and τ is a scalar multiple. We have:

$$\begin{aligned} f_i(x_0 + \vec{v}\tau) &= (x_0 - x_{p_i})^T \, M \, (x_0 - x_{p_i}) + g_i(x_0) \, \vec{v} \, \tau + \vec{v}^T M \, \vec{v} \, \tau^2 \\ &= d_i(x_0) + g_i(x_0) \, \vec{v} \, \tau + \vec{v}^T M \, \vec{v} \, \tau^2 \end{aligned}$$

3.3 Algorithm

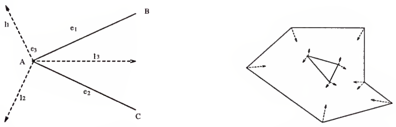
The Voronoi diagram gives us a complete description of the function $t(x)$, that returns the distance of the point x to the closest site in the set S. In particular,

- A Voronoi region (face) is characterized by a single site e_i, the function is given by $t_i(x)$.

- A Voronoi edge is characterized by two sites, e_i and e_j, the edge comprises the set of points where $t_i(x) = t_j(x)$.

- A Voronoi vertex is characterized by 3 or more sites, i, j, k,...m, the vertex satisfies the locus $t_i(x) = t_j(x) = t_k(x) = ...t_m(x)$.

Given an initial Voronoi vertex, and the initial direction of the Voronoi edge emanating out of that vertex, we follow the path traced by this edge to determine the vertex at the other end. Every other site is examined to find the closest site that determines the new vertex. The new vertex is equidistant from three or more sites, every pair of which gives rise to a possible new Voronoi edge. The new Voronoi edges are added to an unexamined edge list. The program terminates when all the edges have been traced.



(a) Voronoi edges of 2 sides of a polygon

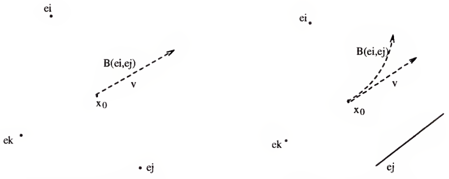
(b) Initialization for a polygon with holes

Figure 3.2. Initialization

3.3.1 Initialization

Given a pair of successive segments of any polygonal region, this gives rise to the following three Voronoi edges, as shown in Figure 3.2a. BA and AC are two successive sides of a polygon. There are three sites corresponding to these two sides; two line sites e_1 and e_2 corresponding to the open line segments BA and AC, and the point site e_3. The three Voronoi edges corresponding to these sites are shown by the dashed lines. l_1 is the bisector of e_1 and e_3, l_2, that of e_2 and e_3, and l_3, the e_1, e_2 bisector. If we are considering the Voronoi diagram of the interior of the polygonal region alone, we are only concerned with those Voronoi edges inside the region. For the outer boundary of the polygon, these are the $(line, line)$ bisectors of every convex pair and the (*point*, *line*) bisectors of every concave pair of successive polygon segments (edges of type l_3, and edges of type l_1 and l_2 respectively). For the inner boundaries (the holes), we perform the reverse. Figure 3.2b shows the complete initialization for a simple case. The vertices of the polygon are also Voronoi vertices. Each of the edges determined in this step are added to the unexamined edge list.

The unexamined edge list holds edges which have been only partly determined. Specifically, the information known, is the starting point and the initial direction along which we must traverse, to determine the other points along the edge. As has been noted earlier, if the Voronoi edge is a (*point*, *point*) or (*line*, *line*) bisector, it will be a straight line, in which case there is only a single terminating Voronoi vertex to be determined. If we have a (*point*, *line*) bisector, the Voronoi edge is a parabolic curve. This curve has to be traced, and intermediate points along it computed, to fully determine the Voronoi edge. The curve tracing step is the topic of the next section.



(a) $B(e_i, e_j)$ is along the direction of $\vec{v}$

(b) $\vec{v}$ is tangent to $B(e_i, e_j)$ at x_0

Figure 3.3. Initial Direction for a Voronoi edge

3.3.2 Curve Tracing

Consider a Voronoi edge E from the edge list. The initial starting point is x_0, which is a Voronoi vertex equidistant from three or more sites, e_i, e_j, $e_k ... e_m$. Assume without loss of generality, that the edge being traced is a bisector of sites e_i and e_j. The initial direction of the bisector is determined upto the linear order, and is given by $\vec{v} = (g_i(x_0) - g_j(x_0))^{\perp}$, where $a^{\perp}$ denotes a unit vector perpendicular to the vector a. For the linear case, the bisector is along the direction of $\vec{v}$. When the bisector is of quadratic order (in the $(point, line)$ case), $\vec{v}$ describes the tangent to the curve at the point x_0. See Figure 3.3.

Now consider the function $f_l(x_0 + \vec{v}\tau)$. The variable τ parameterizes points along the vector $\vec{v}$. Recall that f is a quadratic polynomial in τ that gives the square of the distance from a site e_l to the point $(x_0 + \vec{v}\tau)$.

Let f = max (f_i, f_j). When the bisector is a straight line edge, it is along the direction of $\vec{v}$, and hence in this case $f_i = f_j$. In the nonlinear case, f_i and f_j will not be identical beyond the linear term and f represents the greater of the two. At x_0, $\tau = 0$, and by definition of a Voronoi vertex, we have

$$f_i = f_j = f_k = ...f_m < f_l \qquad \forall\, l \in (S - (e_i, e_j, e_k, ..e_m))$$

We need to trace a path starting at x_0, such that

$$f < f_k, .., f_m, f_l \qquad \forall\, l \in (S - (e_i, e_j, e_k, ..e_m))$$

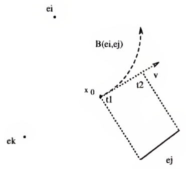


Figure 3.4. Relevant region for e_j: $\tau \in (t1, t2)$

A new vertex is found when we reach a point where

$$f_i = f_j = f_l \text{ for some site } e_l.$$

With respect to every line site e_i is associated a relevant region of a given bisector $\vec{v}$ expressed as an interval of τ, determined by the contact points from the bisector to the endpoints of the site. See Figure 3.4.

We need to ensure that the set of constraints admitted by $f < f_l \; \forall \; l$, is maintained at every point on the bisector of i and j. This is achieved as follows. We first determine the admissible range of τ for each site l. The admissible range is computed as the complement of the inadmissible range of τ, which is defined as the region where $f > f_l$ in the relevant region of e_l. The admissible range, $T^i_{admissible}$ is computed for every site e_i. The intersection of these regions gives us T_{final}. Thus, we have

$$T^i_{admissible} = [0, \infty] - T^i_{inadmissible}$$

$$T_{final} = \cap_{e_i \in S} T^i_{admissible}$$

Note that T_{final} can be a set of intervals, one of which will be $[0, \tau_v]$. This is the interval of interest; 0 corresponds to the initial Voronoi vertex, and τ_v, the τ value for the new vertex. The new vertex is given by

$$x_t = x_0 + \vec{v}\tau_v$$

Let e_l be the site that determined the value of $\tau = \tau_v$. Then at x_t, $f = f_l$. When the bisector of e_i and e_j is a line, the new vertex determined by this procedure, is in fact a Voronoi vertex corresponding to the sites e_i, e_j and e_l, since, for this case $f_i = f_j$ along $\vec{v}$ and at x_t, we have $f_i = f_j = f_l$.

3.3.3 The Nonlinear Case: Tracing Parabolic Edges

In the nonlinear case, x_t is a point on the linear approximation of the bisector. In this case, we employ an iterative technique to converge onto the Voronoi vertex at the other end of the curve. Having determined x_t, a point on the linear approximation of the bisector, we need to move back to a point on the curve itself. It is easy to see that we will intersect the bisector $B(e_i, e_j)$ if we move along the contact vector $-g(x_t)$ towards the the point site. Thus, it is easy to determine x_{t1}, an intermediate point along $B(e_i, e_j)$. See Figure 3.5. In Lemma 2, we prove that this procedure is guaranteed not to overshoot the Voronoi vertex. Namely, we show that the open interval of the bisector $B(e_i, e_j)$ between the points x_0 and x_{t1} contains no Voronoi vertex. Having determined an intermediate point on the bisector, the procedure is repeated using a new value for $\vec{v}$ that is given by $\vec{v} = (g_i(x_{t1}) - g_j(x_{t1}))^{\perp}$. The procedure terminates when we arrive at a point x_{tn} which corresponds to a site e_n, such that $f_i = f_j = f_n$. x_{tn} is the new Voronoi vertex.

We note that the rate of convergence of the iteration is the same as that of Newton's method, since we are approximating an arc by a straight line.

<u>Lemma 2</u> *The open interval of the bisector $B(e_i, e_j)$ between the points x_0 and x_{t1} contains no Voronoi vertex. That is, for this set of points $f_i = f_j < f_k; k \neq i, j$.*

Proof: Consider an arbitrary point x in the open line segment (x_0, x_t). (See Figure 3.6.) If the bisector is nonlinear then one of the sites e_i, e_j will be closer and the other one further from x. Let e_j be the closer site, so that $f_j(x) < f_i(x) = f(x) < f_k(x); k \neq i, j$. Draw a circle of radius $\sqrt{f}$ centered at x. This circle will contain portions of site e_j in its interior, and will touch site e_i and all other sites e_k will lie outside this circle. Shrink this circle such that it continues to touch site e_i and it center continues to lie on the line connecting the point x to the contact point of the circle with e_i until the circle just touches e_j. The center of the circle will move in the direction $-g_i(x)$ and the new center will be the point x_1 which lies on the bisector $B(e_i, e_j)$. The shrunk circle lies in the interior of the original circle and hence all sites $e_k, k \neq i, j$ will continue to lie outside the circle. Hence at the point x_1, $f_i(x_1) = f_j(x_1) < f_k(x_1), k \neq i, j$. Hence x_1 cannot be a Voronoi vertex. Since by choosing all the points in the interval (x_0, x_t) in the open line segment we get every point on the bisector $B(e_i, e_j)$ in the interval $(x_0, x_t 1)$ this result holds for the entire interval. ■

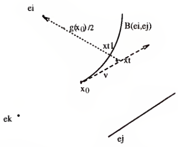


Figure 3.5. Finding an intermediate point on the curve

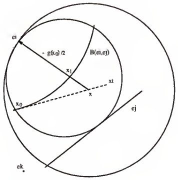


Figure 3.6. Proof of Lemma 2

3.3.4 Increasing Robustness

While the above curve tracing procedure is guaranteed to converge to a Voronoi vertex. In the nonlinear case however, it is possible that as we come closer and closer to the vertex, the very small distances involved in the numerical comparisons give rise to numerical problems that adversely affect robustness. This is particularly true when two Voronoi vertices are extremely close to each other.

The procedure outlined below reduces the number of times we need to employ the curve tracing procedure to obtain parabolic Voronoi edges. The basic idea is to defer the tracing of parabolic Voronoi edges until all (or most) of the Voronoi vertices have been determined. This implies that when we determine the new edges arising out of a newly computed Voronoi Vertex, the nonlinear edges are added to the end of the unexamined edge list and the linear edges are added to the front of the List. Thus most of the linear edges will be examined before the nonlinear ones, and consequently most of the Voronoi vertices will be determined before the nonlinear edges are traced. The truth of the last statement stems from the fact, that there must be at least one linear edge arising out of any Voronoi vertex.

Recall that the unexamined edge list contains Voronoi edges that have not been completely determined. In particular, only the starting vertex and the direction of the edge is known. Thus, if the two endpoints of a given edge, are determined as a result of tracing two different edges that terminate in the respective endpoints, we will have two entries in the unexamined edge list that represent the same Voronoi edge. When a Voronoi vertex is determined as a result of tracing a Voronoi edge, the vertex list is searched to verify if this vertex already exists. If so, it implies there must be another entry in the unexamined edge list corresponding to the edge being traced.

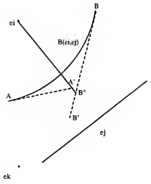
If there are two unexamined parabolic edges that are bisectors of the same pair of sites, each of which start from a different Voronoi vertex, we can avoid the curve tracing procedure, if we can confirm that the two vertices are the two endpoints of a single Voronoi edge. Intermediate points on the curve are determined easily by using the same technique used in the curve tracing procedure.

How often will such a case occur - namely, that the two vertices of a parabolic Voronoi edge be determined before the parabolic edge is traced ? Since there must be at least one linear edge arising out of any Voronoi vertex, most of the vertices of the Voronoi diagram can be determined by tracing the straight line edges before the parabolic edges. Tracing a straight line edge is a much simpler numerical operation than that for tracing parabolic edges. What remains is to devise a procedure to verify that two vertices that are the starting points of two unexamined parabolic edges are the endpoints of a single Voronoi edge. The following Lemma gives us the basis for such a procedure.

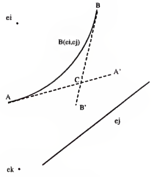
Lemma 3 *Let E_1 and E_2 be two unexamined parabolic edges, corresponding to the same pair of sites e_i and e_j such that E_1 starts from Voronoi vertex A and E_2 from vertex B. Then AB is a single Voronoi (parabolic) edge if the linear approximations from A and B cross each other.*

Proof: Consider Figure 3.7. Subfigures 3.7a, 3.7b and 3.7c depict the various cases that can arise when iterating from A and B simultaneously.

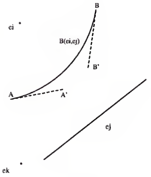
In Case 1 (3.7a), the linear approximations AA' and BB' do not cross, but the line drawn from the point site e_i through A' (respectively B') meets the line BB' (respectively AA') in a point that belongs to the open interval BB' (respectively AA'). Let the line from e_i through A' meet BB' in B''. Then at B'' $f_j < f_i = f < f_k$, for all $k \neq i, j$. From Lemma 2, we know that this relation holds as we move towards e_i. Hence at A', we must have $f_j \leq f_i = f < f_k$. But since A' is the linear approximation of the Voronoi vertex starting from A, we must have $f_j = f_k$ at A' for some site k. Thus the linear approximation from A cannot terminate at A'. Hence Case 1 is impossible, and we must have either Case 2 or Case 3.



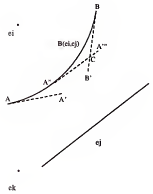
(a) Case 1



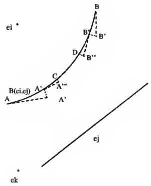
(b) Case 2



(c) Case 3



(d) Case 3a



(e) Case 3b

Figure 3.7. Figure for Lemma 3

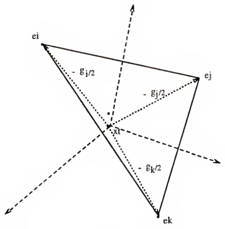
In Case 2, the two linear approximations cross. Lemma 2 guarantees that we cannot overshoot a Voronoi vertex; hence there can be no Voronoi vertex from BB' or AA' along the curve. Hence AB must be a single parabolic edge. It is possible to determine intermediate points on the curve by projecting back from points on the lines AC and BC to the site e_i.

Finally, it is possible to have a situation as depicted in Case 3. Here the two linear approximations do not intersect. In such a case, we need to repeat the procedure starting at one of the points A' or B'. If the new edge starting from A' crosses the one starting from B, as shown in Case 3a, we have verified that AB is a single Voronoi edge, or repeated iterations may converge to two new Voronoi vertices C and D as in Case 3b, and we have found two Voronoi edges AC and BD. ■

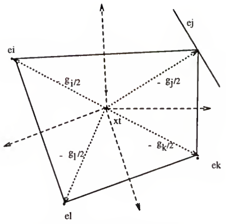
3.3.5 Adding New Edges

Having computed a new Voronoi vertex x_t, we must first perform a check to determine if this vertex has been found before. If this vertex has been computed earlier, it means that there must exist in the edge list, an unexamined edge with starting point x_t. This edge must be deleted from the edge list, since it has now been fully determined. If the new vertex has not been computed before, we proceed to determine all the edges coming out of this vertex.

The first step is to find all the sites equidistant from this vertex. The second step is to determine the new Voronoi edges that emanate out of the new vertex.



(a) 3 co-circular point sites



(b) 4 co-circular sites with 1 line site

Figure 3.8. Determining edges from the convex hull of points of contact

The first step is easy to perform; all sites e_l, which satisfy

$$d_l(x_t) = d_i(x_t) = d_j(x_t)$$

lie on the circle of radius $d_l(x_t)$, centered at the new Voronoi vertex x_t.

The second step is to determine all the edges coming out of this vertex. In general, if we have k sites equidistant from a vertex, there can be $\begin{pmatrix} k \\ 2 \end{pmatrix}$ pairs of possible edges, but all of these will not be Voronoi edges. From the vector $g(x_t)$, for each site e_i, we can determine the point of contact of $-g_i/2$ with e_i. If we denote the points of contact as y_i, the convex hull of the points, y_i, determines which pairs of sites correspond to Voronoi edges. Every convex hull edge, gives a pair of sites that correspond to a Voronoi edge. Thus if we have k equidistant sites from the vertex x_t, there are exactly k Voronoi edges. See Figure 3.8.

One of these k edges is the edge which was just traced. The other $k-1$ edges are appended to the unexamined edge list. The head of the edge list is then examined and the procedure repeats until the List is empty.

Due to the incremental nature of the edge tracing algorithm, we do not need to make any assumptions about points in general position. Hence, co-circularity of more than 3 sites can be handled easily.

3.4 Time Complexity

Let the number of sites in the input be n_p, the number of Voronoi edges be n_e and the number of Voronoi vertices be n_v.

The initialization step is of the order of the number of sites in the input, since a single pass over the input suffices to create the initial edges and append them to the unexamined edge list. This step takes $O(n_p)$ time.

Each edge that is added to the diagram in the edge tracing step requires us to examine each site to determine the closest site. Also, we need to search the list of current Voronoi vertices to check for the existence of a new vertex. Hence, we need $O(n_p+n_v)$ time for each new edge found. The total time required for the algorithm is therefore $O(n_p n_e + n_v n_e)$. By Euler's formula the number of Voronoi edges is $2n_p - 3$ and the number of Voronoi vertices $n_p - 2$. Hence the time complexity is $O(n_p^2)$.

CHAPTER 4
SKELETONS FROM VORONOI DIAGRAMS

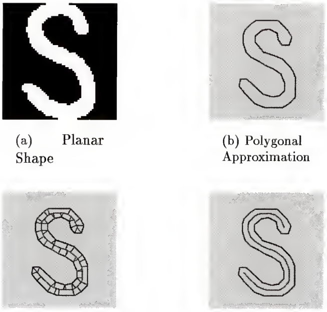
In this section, we describe our technique for obtaining skeletons of character patterns that are derived from Voronoi diagrams.

4.1 Preprocessing: Computing a Polygonal Approximation

In order to compute the Voronoi diagram of a shape (Figure 4.7a), we first segment the shape's boundary, and derive a polygonal approximation of its bounding contour. In our implementation, we used the zero crossings, obtained from an image filtered with a Laplacian of a Gaussian [30] to obtain a bounding contour. The advantage of this technique is that it guarantees closed contours, which ensures that a contour tracing algorithm such as chain coding will converge rapidly. After obtaining a chain coded bounding contour, a polygonal approximation is computed. The input to the polygonal approximation routine is the chain coded bounding contour (Figure 4.7b). The goal of the approximation process is to ensure that significant changes in curvature are retained, while minor distortions that could result in noise spurs in the skeleton representation are smoothed over. This limits the occurrence of redundant edges in each representation. The most accurate polygon that can be derived from this contour will be one that retains every vertex which reflect a non-zero change in curvature. Thus pixels that constitue a straight line in the chain code are segments of the polygonal approximation (Figure 4.7b). However, this representation is characterized by edges of extremely small size which could lead to redundant skeletal edges See Figure 4.7c. Thus we refine the approximation process as follows. We introduce three parameters, $PIXEL\text{-}THRESH$, $ANGLE\text{-}THRESH$ and $STRAIGHT\text{-}THRESH$, that serve to control the accuracy of the approximation process. The parameter $PIXEL\text{-}THRESH$ controls the minimum number of pixels that comprise a segment. As we add pixels to a segment, we keep track of the angle between successive segments, averaged over all pixels comprising this segment. If the current segment consists of pixels $p_1..p_(n-1)$, then pixel p_n is added to the current segment if the angle made between p_{n-1} and p_n differs from the mean by an amount less than $ANGLE\text{-}THRESH$. In the event that the current angle differs from the mean by an amount greater than $ANGLE\text{-}THRESH$, p_n becomes a vertex of the polygonal approximation. The averaging operation is bypassed for groups of pixels of length greater than $STRAIGHT\text{-}THRESH$, that lie along a straight line. The end-points of such groups serve as vertices of the final polygon. Typical values chosen for our application were: $PIXEL\text{-}THRESH = 4$, $STRAIGHT\text{-}THRESH = 4$ and $ANGLE\text{-}THRESH = 30$. These values were determined experimentally. After we introduce our pruning technique in the next section, we will discuss the effects of the polygonal approximation on the skeletonization process.

4.2 Computing the Voronoi Diagram and Extracting Skeletons

Given the vertices for a polygonal approximation, we compute the Voronoi diagram using the algorithm described earlier in Chapter 3. The medial axis [26] of a



(a) Planar Shape

(b) Polygonal Approximation

(c) Voronoi Diagram

(d) Pruned Skeleton

Figure 4.1. Voronoi Skeletonization of Planar Shapes.

polygonal shape can be derived from its Voronoi diagram by deleting those edges that arise from concave vertices of the polygon (see Chapter 3.) The data structure created to store Voronoi edges contains the information required to identify these edges. Hence it is easy to identify (and delete) edges that emerged from concave vertices of a polygon. However, resultant medial axes are characterized by a very large number of edges, as every vertex of the polygonal approximation (both concave and convex) gives rise to a Voronoi edge. Many of these Voronoi edges are redundant; i.e. they do not provide any additional structural information about the planar shape. Hence a pruning technique is required to delete redundant edges.

4.3 The Pruning Operator

As shown in Figure 4.1c, the vertices of the initial polygonal approximation of a shape lead to a large number of edges that do not contribute to overall shape information. In a Voronoi diagram every Voronoi edge is a bisector of two sites on the boundary of a polygon. In particular, the Voronoi edges at the vertices of a polygon are bisectors of adjacent sites. If the sides of a polygon are numbered in counterclockwise order along the boundary, we can define the adjacency of a Voronoi edge as follows. Let E be a Voronoi edge that is a bisector of two sites numbered i and j. Then $Adjacency(E) = |i - j|$. We observe that Voronoi edges that lie deep inside an object have higher adjacency than those on the perimeter. Furthermore segments that describe "global" topological (symmetry) relations are bisectors of high adjacency. This fact provides a heuristic for filtering out unimportant segments. We discard edges that are of lower adjacency than some preset threshold. Later in this section, we describe how we choose the pruning threshold. Since the adjacency information is implicitly contained in our edge data structure (see Chapter 3), no additional post-processing is required to obtain this reduced efficient set of edges.

We note that the authors of [38] use a similar (but not identical) technique for pruning the edges of Voronoi diagrams. The problem in their case is more critical, since they start out with a set of raster crack end-points along the boundary of an object and hence had an extremely large number of Voronoi edges to process.

Figure 4.1 shows the result of the three step procedure described above on a planar shape of the character 'S'. Figure 4.2 presents additional examples of skeletons derived using our skeletonization technique.

4.4 Preserving Connectivity

As described in Chapter 2, one of the advantages of skeletons derived from Voronoi diagrams is that we are guaranteed connected skeletons. However, we need to consider the effect of the pruning step on connectivity. Another issue to be resolved, is to determine an optimal threshold for deleting the maximum number of redundant skeletal edges without losing connectivity. Unfortunately, it is not possible to determine a threshold value a priori and apply it to a large class of objects. However, by constraining the threshold value to 1 (retaining only those edges that are of adjacency greater than 1), we can prove that the resultant skeleton remains connected.

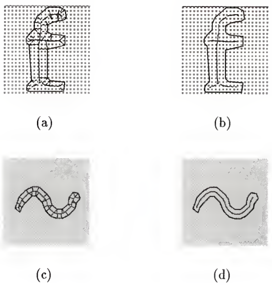


Figure 4.2. (a),(c) Sample Voronoi Diagrams, (b), (d) Corresponding pruned skeletons

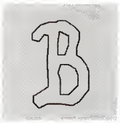
Lemma 4 Pruning Voronoi edges of adjacency 1, is guaranteed to preserve connectivity.

Proof: The Voronoi diagram of a polygon is a Planar Straight Line Graph [48], which by definition, is connected. For the removal of an edge to result in a disconnected graph, the edge must be a "bridge". Namely, the edge must be part of every path between any two vertices of the graph. It is easy to see that no edge of adjacency 1 can be a bridge, since (by definition of adjacency) every edge of adjacency 1 terminates at a vertex of a polygon. Hence removal of all edges of adjacency 1 is guaranteed to preserve connectivity of the resulting skeleton. ■

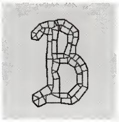
4.5 Skeletonizing Shapes with Holes

One of the attractive features of our method is that it handles shapes with and without holes in a uniform manner. Figure 4.3 shows an example of our approach

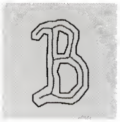
(a) Planar Shape



(b) Polygonal Approximation



(c) Voronoi Diagram



(d) Pruned Skeleton

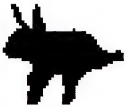
Figure 4.3. Skeletons of Character shapes with holes

applied on a object with holes. No additional initialization is required. However, when carrying out the pruning step, care must be taken to define the adjacency information correctly, in order that the last numbered site on the outer boundary and the first numbered site on the next hole are not considered adjacent. Shapes with multiple holes are handled similarly.

4.6 Stability and Robustness

A major problem of the Symmetric Axis Transform, is its sensitivity to noise. As has been noted above, there exists a skeletal edge associated with every convex vertex. If the bounding contour is extremely jagged, this can lead to a large number of local symmetric axes that are redundant. Many approaches have been proposed in the past to reduce the effects of noise on the skeletal representation. These approaches take the form of either smoothing the contour, or using a pruning approach to eliminate some of the redundant edges. Pizer et al. [42] smooth the contour, enroute to achieving a hierarchical shape description using the skeleton. Blum [6] associates a weight with each symmetric axis and then applies a thresholding operator to delete those axes with low weights. Rom and Medioni [44] approximate the bounding contour with cubic splines. The approximation of a noisy contour by a cubic spline limits the creation of spurious symmetric axes. They then break the shape into its constituent parts at negative curvature extrema to compute a hierarchical symmetric axis representation. Brandt and Algazi [9] and Ogniewicz [38] both compute the discrete Voronoi skeleton by sampling the bounding contour. Pruning techniques are then applied to reduce the number of Voronoi edges.

A problem with pruning techniques in general, is that they are based on heuristics which decide whether an edge is redundant or relevant. Hence, there are always cases where over–pruning or under–pruning results, that cause either relevant edges to be deleted, or redundant edges to be retained. An application independent solution does not seem to be readily available. Part of the problem is the rather fuzzy definition of a redundant edge, since this may be application dependent. For example, in an industrial application that checks for defects in manufactured parts [8, 10], the skeletal edges associated with minor distortions should be retained. On the other hand, in character recognition applications, minor distortions along the contour could reflect variations in individual handwriting style and it is desirable to have a skeletal representation that is invariant to such distortions.


(a) Planar Shape

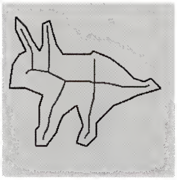
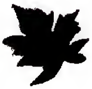

(b) Voronoi Skeleton

Figure 4.4. Skeleton of a animal shape.


(a) Planar Shape

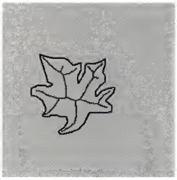

(b) Voronoi Skeleton

Figure 4.5. Skeleton of a leaf shape.

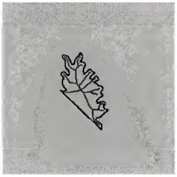


Figure 4.6. Polygons with jagged edges

In a more recent paper, Ogniewicz [37] presents a multiscale representation based on discrete Voronoi skeletons that combines a pruning technique along with smoothing the bounding contour at various scales. This enables the author to build a hierarchical skeleton-space, wherein pruning and scaling parameters can be automatically selected. The advantage of this scheme is that it works well for shapes with nonuniformly scaled boundaries. However, in some cases oversmoothing or undersmoothing may occur, that causes either redundant edges to persist, or relevant edges to be deleted.

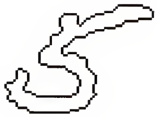
The pruning operator presented in this dissertation is based on the simple notion of adjacency as defined in an earlier section. Together with the constraint restricting the pruning threshold to be one, a simple pruning scheme results. However, this scheme may not always give the best results. In particular, for an irregularly scaled planar shape, (one that has jagged edges in one region and fairly straight edges in another, as in Figure 4.6), the resultant representation will have redundant edges. Further increasing the threshold to delete the redundant edges would result in deleting relevant edges as well. In applications where such shapes abound, a more sophisticated pruning scheme such as the one in [6] or [38], would need to be applied. In character recognition applications however, we found that the polygonal approximation gave us fairly regular polygons which allowed us to compute skeletons with almost no redundant edges. See Figures 4.5 and 4.4.

The quality of the skeletons obtained by the techniques presented in this section are closely related to the polygonal approximation of the planar shapes. The scale of the filter used to obtain the bounding contour and the paramters of the polygonal approximation will impact the accuracy of the resulting polygon. Hence, it is of interest to see how stable the process is, to variations in the approximation process. As mentioned in an earlier section, extremely fine approximations result in very small length edges of the polygon that lead to redundant skeletal edges. This can be noted in Figure 4.7c. The planar shape shown in Figure 4.7a is common to the three experiments that follow. In all the experiments, the scale of the filter was given by $\sigma = 1.0$. In the first experiment, shown in Figure 4.8, skeletal representations for increasing values of *PIXEL-THRESH* were computed. The parameters *STRAIGHT-THRESH* and *ANGLE-THRESH* were kept constant. In the second and third experiment, we varied *STRAIGHT-THRESH* and *ANGLE-THRESH* respectively, keeping the other two parameters constant. The resulting skeletal representations are shown in Figure 4.9 and Figure 4.10. As seen in the figures, except for very low values of each parameter, redundant edges are largely eliminated. Typical values chosen for our application were, *PIXEL-THRESH* = 4, *ANGLE-THRESH* = 23 and *STRAIGHT-THRESH* = 4.

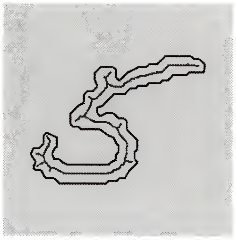
4.7 Performance Evaluation

In this section, we briefly look at the time complexity of the entire skeletonization process. In the discussion that follows, let N denote the size of a square image, ($N = M * M$), where M is the length of each row and column. Let B represent the average size of the bounding contour of each planar shape. In Section 2, we saw that the time complexity of the Voronoi diagram algorithm is $O(B^2)$, where B is the number of sites

(a) Planar Shape

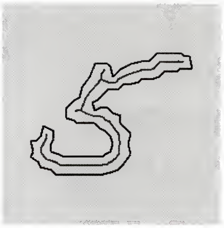


(b) Output ofsegmentation

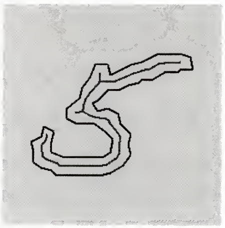


(c)

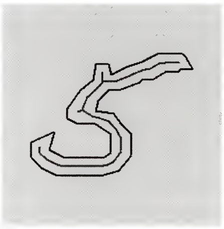
Figure 4.7. Perfect polygonization can lead to redundant skeletal edges.



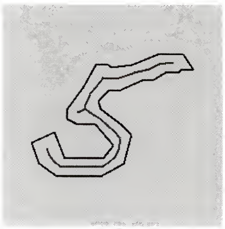
(a) *PIXEL-THRESH* = 2



(b) *PIXEL-THRESH* = 4

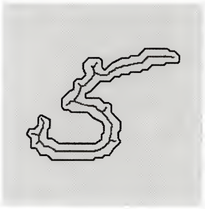


(c) *PIXEL-THRESH* = 6

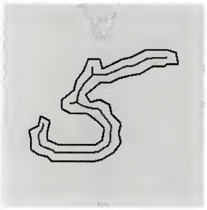


(d) *PIXEL-THRESH* = 10

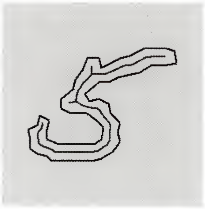
Figure 4.8. Polygonal Approximation for varying values of $PIXEL\text{-}THRESH$. $ANGLE\text{-}THRESH = 23$ and $STRAIGHT\text{-}THRESH = 3$.



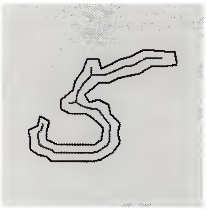
(a)
STRAIGHT-THRESH $= 0$



(b)
STRAIGHT-THRESH $= 4$

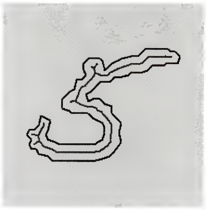


(c)
STRAIGHT-THRESH $= 8$

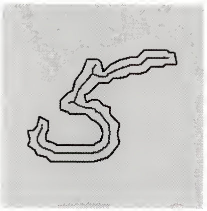


(d) *STRAIGHT-THRESH* $= 12$

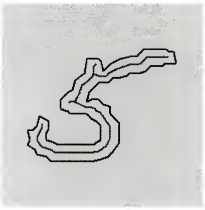
Figure 4.9. Polygonal Approximation for varying values of *STRAIGHT-THRESH*. *PIXEL-THRESH* $= 4$ and *ANGLE-THRESH* $= 23$.



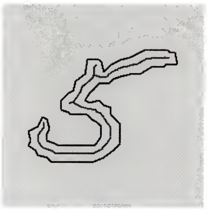
(a) *ANGLE-THRESH* = 10



(b) *ANGLE-THRESH* = 23



(c) *ANGLE-THRESH* = 35



(d) *ANGLE-THRESH* = 52

Figure 4.10. Polygonal Approximation for varying values of *ANGLE-THRESH*. *PIXEL-THRESH* = 2 and *STRAIGHT-THRESH* = 4.

Table 4.1. Performance Evaluation: Running times reported per 64x64 character image.

	Time Complexity	Actual Running Time
Segmentation	$O(N \log N)$	2.26 sec
Polygonization	$O(B)$	0.10 sec
Voronoi Skeleton	$O(B^2)$	0.81 sec

on the bounding contour. Existing skeletonization algorithms have time complexities that are a function of N alone, and hence valid comparisons are difficult. In practice, we have observed that in spite of the quadratic complexity, the algorithm is efficient in practice, since the number of boundary points B is generally an order of magnitude less than N. For completeness, however, the preprocessing steps must also be taken into account. The table below summarizes both the theoretical time complexity of each stage, as well as the actual execution time taken on a Sparc 10 computer to compute the skeleton of each character shape. The timing measurements are reported for each character of size 64x64, averaged over a total of over 1,350 distinct character samples. Note that the Voronoi construction processing time accounts for only 25% of the total time required for skeletonization.

In the next section we show that Voronoi skeletons computed by the procedure described above, provide a more efficient representation compared to existing skeletonization algorithms.

4.8 Comparison of Other Skeletonization Techniques

Our approach exploits the fact that the medial axis of a polygonal shape is implicitly contained in its Voronoi diagram. This fact immediately ensures (a) the computed skeleton lies in $\mathcal{R}^2$, (Euclidean metrics), and (b) that connectivity is guaranteed. We compute the continuous medial axis as opposed to a discrete approximation. In the following paragraphs we show that these properties provide a significant advantage over traditional skeletonization techniques.

Skeletons computed by thinning algorithms are constrained by 4, 8 or 6 connectivity of a discrete grid in which the object shapes are embedded [13]. The typical drawback of such approaches is the loss of Euclidean metrics.

An alternative approach to skeletonization involves ridge following techniques based on distance maps computed from object shapes. The quality of the resulting skeletons is critically dependent on the metric used to derive the distance map. Non-Euclidean metrics (city-block distance, chessboard distance among others), lead to simple skeletonization algorithms, but can result in inaccuracy of upto 40% with respect to Euclidean distances [28]. While methods to compute Euclidean [12] or quasi-Euclidean [33] distance maps exist, these methods do not guarantee connectivity. Gaps occur due to the discrete domain on which pattern shapes are embedded, and are usually filled in by postprocessing steps.

More recently, new skeletonization algorithms, [9], [38], [28] have appeared in the literature that are marked improvements over the traditional techniques described above. In the sequel, we briefly describe these methods, and compare them with our approach.

In the paper by Brandt et al. [9], the authors compute the discrete Voronoi skeleton by sampling the bounding contour. By adopting an input model where the shape has been sampled along its boundary, and where the shape is known to be r-regular, the accuracy of the resulting skeleton can be predicted from the density of the sampling. The authors derive a bound for the regeneration error. This is the difference between the original planar shape and the shape regenerated from the discrete skeleton, which is shown to be less than 1 pixel.

Ogniwiecz [38, 37], employs a similar technique of computing discrete Voronoi skeletons. The difference lies in the pruning heuristics used and the use of a hierarchic skeleton-space that allows for representing shapes with irregularly scaled contours.

The discrete point Voronoi diagram results in skeletons that well approximate the continuous skeletons only if the sampling is sufficiently dense. Typical examples in the above papers use anywhere between 1000 and 10000 points for a given component. This large number leads to a equivalently large number of Voronoi edges, most of which are redundant. A more critical drawback of the discrete Voronoi diagram approach is that it is not easily applicable to objects with holes. In particular, it is not possible to decide a priori, the number of sampled points on an object, necessary to ensure that the Voronoi edges between the inner and outer boundaries will be computed.

In our approach, polygons with and without holes are handled in a uniform manner. Polygons are defined in terms of line segments, and are therefore well defined at every point $p(x, y) \in \mathcal{R}^2$ along the boundary. This overcomes the drawbacks of discretizing the bounding contour as in the techniques discussed above [38, 9]. Thus a simple pruning step, that involves no postprocessing suffices to eliminate almost all spurious edges. However, our pruning operator suffers from the drawbacks described in the previous section. Since the continuous skeleton is computed, the regenerated shape will correspond exactly to the input polygon. Therefore, in our case, the regeneration error is bound by the accuracy of the polygonal approximation process.

Leymarie et. al [28] employ the snake model (an active contour model), to compute skeletons. In their technique, initial control points are defined for the snake at positive curvature extrema of a shape's bounding contour. A dynamic grassfire is then simulated by snake propagation. This technique maintains correct Euclidean metrics by computing the Euclidean distance map, while the nature of the contour model ensures that connectivity is maintained. This method is also noise sensitive. A very ragged boundary implies a large number of curvature extrema leading to redundant skeleton edges; in this case other criterion need to be applied to reduce the number of initial control points. In addition, a special case arises when the bounding contour includes a circular arc whose center (a) lies within the object and (b) is an end-point of a skeleton branch. In such a case, additional control points need to be defined to ensure accurate skeleton computation. In contrast, our technique offers an uniform approach to skeletonization that is independent of the shapes topology.

An implicit advantage offered by our method, is that a graph representation of the skeleton is immediately available. Recall that the Voronoi diagram of a polygon is a planar straight line graph, and that by Lemma 4, the pruning step maintains connectivity. Skeletonization methods that compute the skeleton as a pixel map (as compared to an edge map representation provided by our method), require an intermediate vectorization step before a graph representation may be obtained. In the context of character recognition systems, our technique is amenable to the simple extraction of structural features such as end-points and junctions by a simple traversal of the graph comprising a skeleton.

In summary, we have shown that Voronoi skeletons are shape descriptors which overcome some of the disadvantages of existing skeletonization techniques. In the next chapter we present results of our character recognition system based on this novel representation.

CHAPTER 5
CHARACTER RECOGNITION

Optical Character Recognition (OCR) finds applications in postal automation, in document understanding, and recently, in pen-based computing. While over three decades of research effort has focussed on various facets of this wide field, much research continues today since the OCR problem remains unsolved. In the sequel, we will outline the various subproblems within the larger context of OCR and discuss the current state of the art and the specific problems that remain to be solved.

OCR techniques can be broadly classified into on-line and off-line recognition techniques. On-line methods refer to those applications where characters are recognized as they are written, as for example in the case of a pen tablet. Off-line systems are designed to recognize characters already in printed form that have been converted to image format by a digital scanning process. Off-line systems are further subdivided into machine printed recognition techniques, handprinted character recognition and cursive script recognition. A majority of the existing commercial OCR systems have been designed to recognize machine printed characters. The parameters here, are the font recognition capability, (whether single or mulit font), the restrictions on the document layout etc. Since this dissertation does not address the machine printed recognition problem, we do not discuss this topic in greater detail. The interested reader is referred to [11, 36, 19] for details on the state of the art in machine printed character recognition.

While some effort in the handprinted character recognition problem has concentrated on cursive script recognition [46], a majority of recent research effort (and particularly academic research effort) has focussed on isolated handprinted character recognition. Handwritten character recognition is an important subproblem within the broader field of Optical Character Recognition (OCR). While many researchers have successfully provided partial solutions, the problem of handwritten character recognition remains unsolved. Specifically, more reliable methods need to be found before human performance in recognizing characters can be matched. Readers are referred to [32] for a historical review of OCR, and to [52, 51] for discussions on handprinted character recognition. We now look at the steps involved in a character recognition system and existing techniques employed in each of these steps.

5.1 Character Recognition Systems

As described in the introduction to this dissertation, a general image analysis system goes through the steps of preprocessing, shape extraction, image representation and recognition/interpretation. In OCR systems, the preprocessing steps include image acquisition and segmentation, where the document is converted to digital format by a scanner and individual characters are isolated by a segmentation process. The segmentation of characters is a complex problem due to the large degree of variance in the way documents are created, and differences in handwriting styles (leading to differences in the spacing between characters and across rows of characters). In this study, we do not focus on the segmentation issue The input to our system is database of characters that has been segmented in a preprocessing step. The interested reader is referred to [47, 21] for discussion of, and solutions to the segmentation problem. Segmented characters are generally preprocessed by one or more of the following: a) Shape preserving size normalization, b) Image smoothing to eliminate noisy contours, c) Rotation and Translation normalization.

The second stage is generally regarded as the most important step in character recognition systems, namely feature extraction. The goal here is to extract a "useful" representation from the segmented character by eliminating redundancy and reducing the dimensionality of the representation. In the context of OCR, the requirement for features extracted are that they possess high inter-class variance and low intra-class variance. In other words, the features extracted should be relatively invariant to variations across distinct instances of the same character pattern, while being sufficiently distinct across different characters. Scale and translation invariant features are also desirable. Feature extraction can be divided into two parts: (a) defining distinctive features of characters, and (b) extracting features once they have been defined. Part (a) of this problem remains open. That existing character recognition systems have yet to match human performance, can in part be ascribed to the intractable nature of defining "distinctive characteristics" for character patterns. In [51], a broad taxonomy of feature extraction approaches is discussed. These include global analysis (techniques such as template matching, measurement densities of points, moments and mathematical transforms) and structural analysis (methods that extract loops, endpoints, junctions and arcs from the contour or skeleton of a character). The latter class is concerned with capturing the essential shape of each character.

Classification of character patterns is the final step in the recognition process. The techniques in use are of two types: adaptive learning-based and rule-based methods. Rule-based methods, sometimes referred to as syntactic or structural methods [32, 52], classify character patterns using a set of rules that are derived from the character's structure. Typical features used are height and angles of strokes, concavities, number of holes, end-points and junctions. The most commonly used adaptive learning techniques are those that use neural networks. The principal difference between neural network methods and rule-based methods is that the former attempts to simulate intelligent behavior by using adaptive learning and the latter uses logical symbol manipulation.

5.2 Errors in Classification

In any pattern classification problem, there exists a set of N patterns belonging to n distinct classes, and the goal is to determine which class an instance of an unknown pattern x belongs to. An error occurs when a pattern x_i belonging to class i is classified incorrectly as belonging to a class j. Depending on the classification technique used, a confidence level can be associated with each classification. This allows for the partition of errors into rejection and substitution errors based on a threshold for the confidence level. Classifications with low confidence values can be rejected, thus causing a rejection error. The pattern is not assigned to any class and may be classified manually. A misclassification with a high confidence value is termed a substitution error. This is a more serious error as the system is now wrongly classifying a pattern rather than rejecting it entirely. The goal in modern day character recognition systems is to strive towards a zero-substitution rate.

5.3 Training and Testing Databases

Most prototype character recognition systems are tested on databases of characters that are split into training and testing sets. Typical partitions of data sets into 66% training set size and 33% testing set size are typical. However, the size of the database and the quality of the characters used vary considerably. A common parameter that varies considerably is the constrained nature of the characters. Some training systems impose constraints on the writers to ensure easier recognition. Typical constraints include having slashed zeroes, only open (or only closed) fours, uncrossed sevens and so on. Results on constrained characters are less meaningful as such conditions are rarely prevalent in practice. Another parameter that varies is the nature of the training and testing data sets. Some systems include the same samples in both the training and testing sets; such results have less value than those that operate on distinct training and testing sets. This makes it very difficult to compare distinct character recognition systems since the training parameters vary greatly. Suen [52] suggests that any character recognition system that merits serious consideration should have:

1. unconstrained handwritten characters

2. relatively large number of samples

3. relatively large number of writers

4. distinct training and testing sets

5. high recognition and low susbstitution rates

In the next chapter, we present our approach to the character recognition problem and discuss our results.

CHAPTER 6
CHARACTER RECOGNITION WITH VORONOI SKELETONS

In this chapter, we describe our character recognition system and discuss our recogntion results.

6.1 Overview of the System

Given a segmented character pattern, its Voronoi skeleton is first computed using the skeletonization technique described in Chapter 3. Recall that this involves polygonal approximation of the character shape, computation of the Voronoi diagram, and the application of the pruning operator to delete redundant edges. The pruned Voronoi skeleton serves as the feature descriptor for our character recognition system. We choose a back propogation conjugate gradient neural network, described in the next section to classify the character patterns. The classifications are reported with a confidence level ranging from 0 to 1, thus allowing us to separate errors into substitution and rejection errors.

The neural network requires every input pattern to be of constant size. Recall that the Voronoi skeleton algorithm outputs a set of connected edges. Since the number of edges will vary across distinct instances of characters, we need to transform this set of edges into an input pattern of uniform size. The edge set representing the skeleton of each input pattern is therefore rendered onto a binary image of size 16 by 16. The binary skeletons consisting of 256 input nodes is then used to drive the neural network with distinct training and testing samples from the patterns in our database.

We tested our system on two distinct databases. As a first step, we used a constrained database to test the feasibility of our system. In accordance with the conclusions of the last chapter, we then focussed on a larger unconstrained standard database from the National Institute of Standards and Technology (NIST). In the next few sections we describe our results with each of the two databases.

6.2 Neural Network Topology

We employed a conjugate gradient method to train a neural network [20] containing at most three hidden nodes and upto 52 output nodes. The number of hidden nodes was determined experimentally by varying the number of hidden nodes and identifying the best results. Table 6.3 gives the results for testing a subset of the NIST database for varying number of hidden nodes.

For the first database, the input patterns were preclassified according to their Euler number (the number of connected components minus the number of holes.) This topological sorting step was performed primarily to reduce training time. We found that training three smaller neural networks (for shapes with 0, 1 and 2 holes respectively), took considerably less time than training a single more complex network. We note that classifying characters according to the topology is justified on account of the constrained nature of this database and the quality of our characters (no gaps.) When dealing with databases of poorer quality, it may not be reasonable

[41, 4] to attempt such preclassification without additional preprocessing. No such preclassification was attempted with the NIST database.

6.3 Results with Database 1

In order to both evaluate and compare the performance of our feature extraction and representation methods, we exercised a large database of handprinted characters previously studied [22]. The database consisted of 52 distinct patterns containing over 10,000 samples, collected from 17 different writers. Table 6.1 summarizes the distribution of patterns in our database.

The characters included were well defined. To avoid ambiguity, authors were asked to slash the numeral '0', to print '1' without serifs and to place horizontal bars on the letter 'I'. In addition, only upper-case alphabets were considered, and the numeral '4' was required to be closed. Sample characters are shown in Figure 6.1.

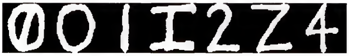


Figure 6.1. Samples patterns for '0', 'O', '1', 'I', '2', 'Z' and '4'.

Table 6.1. Distribution for database of sample patterns.

Patterns	Number of Distinct Patterns	Number of Samples		
		Training	Testing	Total
0-9	10	1,360	680	2,040
Graphics	16	2,176	1,088	3,264
A-Z	26	3,536	1,768	5,304
All Above	52	7,072	3,536	10,608

Sample patterns were collected on forms and were digitized at 300dpi/8bit resolution. Each sample was then normalized in size to 64 x 64 before processing [22].

Table 6.2. Performance evaluation for three classification cases: [0-9], [A-Z] and [A-Z, Graphics, 0-9] for Voronoi skeleton representation.

Classification Cases	Number of Errors	Error Rate	Classification Rate
0-9	7	1.02%	98.98 %
A-Z	25	1.41%	98.59%
A-Z, Graphics, 0-9	83	2.34%	97.66%

We trained the network with three subsets of the database: (a) Numeric characters 0-9, (b) Alphabetic characters A-Z and (c) the entire database of 52 patterns (A-Z, 0-9, and 16 graphic characters). Table 6.2 summarizes our existing experimental results. As shown, results for recognizing numerals and characters were very reliable; correct recognition rates of 98.98% and 98.59% were observed respectively.

6.4 Results with NIST Data

We considered only numeric characters from the NIST special database of hand-printed segmented characters. This database contains 2,100 forms scanned at 12 dots per millimeter binary. Each character has been segmented and placed into individual 128 by 128 pixel binary images. We used a small subset of 60 forms for the current study. The data set is unbalanced. The accompanying table gives the frequency of occurrence of each class.

Table 6.3. Performance evaluation for varying number of hidden nodes. Training size= 11413. Testing size= 500. Rejection threshold= 0.99.

Number of Hidden Nodes	Raw Errors	Raw Error Rate	Number of Rejections	Number of Substitutions	Reliability Rate
1	51	10.2	78	24	5.68
2	44	8.8	87	19	4.6
3	43	8.6	68	19	4.39
4	44	8.8	66	20	4.6
5	46	9.2	68	26	6.01
6	45	9.0	65	20	4.59
7	46	9.2	67	21	4.85
8	50	10.0	67	26	6.0
9	50	10.0	81	25	5.96
10	39	7.8	72	20	4.67

The numerals are unconstrained, with a large degree of variation across samples in a given class. Both open and closed '4's are observed. The numeral '2' occurs with and without loops, and disconnected numerals (mostly the numeral '5') account for about 2% of the samples observed. Figure 6.2 shows some of the samples.

Table 6.4. Frequency of each numeral in NIST data.

	0	1	2	3	4	5	6	7	8	9
Training	460	497	400	435	430	377	436	466	404	424
Testing	202	225	176	193	182	159	188	205	172	178
Total	662	722	576	628	612	536	624	671	576	602

6.4.1 Experiment 1: Determining Optimal Number of Hidden Nodes

In our first experiment, we trained the neural network with varying number of hidden nodes to determine the best configuration. The number of training samples

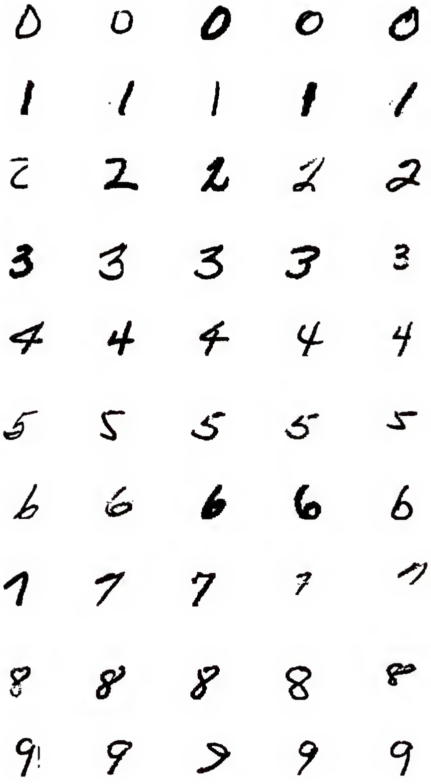


Figure 6.2. Numeral Samples.

Table 6.5. Experiment 2: Performance evaluation for classification of [0-9] for Voronoi skeleton representation on NIST data.

Classification Cases	Substitution Errors	Rejection Errors	Error Rate	Reliability Rate
0-9	72	53	4.31%	95.69%

was 1013, while the number of testing samples was 500. The number of hidden nodes was varied from 1 to 10, and the results tabulated. Table 6.3 summarizes the results. Two columns of error rates are shown. The first is the raw error rate, which is computed before any characters are rejected. The second is the reliability rate which is computed after rejecting characters that were classified with a confidence value of less than 0.99. We obtained best results with 3 hidden nodes and this value is used in the rest of the experiments.

6.4.2 Experiment 2: Single Neural Network Topology

In the next experiment we trained a single neural network [20] containing three hidden nodes and 10 output nodes. The network was trained with 4329 training samples and tested on 1880 samples. Classifications with a confidence value of less than 0.99 were rejected. Table 6.5 summarizes our results. The reliability rate is defined as $Reliability\ \ rate = Num\ \ of\ \ Substitutions/(Total\ \ Num - Num\ \ of\ \ Rejections)$.

6.4.3 Experiment 3: Multiple Neural Network Topology

In our second experiment, we attempted to reduce the number of substitution errors by the following strategy. Rather than having a single neural network learn all the 10 patterns, we trained 10 distinct networks. Each network was trained to recognize a single pattern. For example, the neural network for the numeral 1 was a 256/3/2 network (that is 256 input nodes, 3 hidden nodes and 2 output nodes). The entire dataset was divided into 2 classes; one class containing all the 1 samples, and the second class containing the rest of the data. The networks for the other numerals were trained similarly. There are two advantages to having multiple networks rather than a single one.

1. Since we have 10 distinct networks, they can be trained in parallel on distributed workstations. This reduces the training time.

2. There are fewer number of substitution errors. This is because for a pattern to be termed a substitution error, it has to be misclassifed with high confidence in 2 networks. Consider for example an instance of the number 2, p_2. For p_2 to be misclassified as a 7, the following must hold: p_2 must be misclassifed with high confidence in the 2–network, and must be classifed with high confidence as a 7 in the 7–network. The results support the hypothesis that this pair of events occurs rarely. The result is that we have a high rejection rate, but a very low substitution rate. The results for this experiment are shown in Table 2.

6.5 Comparison of Performance

In the context of handwritten character recognition, it is difficult to exactly compare distinct approaches. There are a large number of variables that can bias results. These include the database size, the method of partitioning training and testing data sets, the kinds of errors reported (substitution and rejection errors), the

Table 6.6. Experiment 3: Performance evaluation for classification of [0-9] for Voronoi skeleton representation on NIST data.

Classification Cases	Substitution Errors	Rejection Errors	Error Rate	Reliability Rate
0-9	32	214	1.92%	98.08%

quality of characters in the database, the nature of the character set (constrained or unconstrained) and so on. Researchers evaluating character recognition systems and results reported in the literature must therefore carefully examine the specific parameters of a character recognition system rather than merely compare percentages of accurate classification.

The NIST database is useful in this regard, since it provides a common ground for researchers to compare results. In addition, the database is characterized by

- unconstrained handwritten numerals
- very large number of samples
- large number of writers

The percentage of correct classification we obtain is comparable to those obtained by some of the leading researchers in this area. In [35], the authors extract structural features from the skeleton and uses 11 specialized modules to come to a decision on the identity of patterns. The authors of [23] also use primitives derived from skeletons for recognition. Both of the last two methods were tested on the same database (US ZIP code database of CENPARMI (Concordia University); the training and testing sets consisted of 4000 and 2000 numerals respectively. Correct classification rates of 86.05% [35] and 93.1% [23] were observed. In [51], a multi-expert system is described which combines the results of four techniques. These include the last two methods described above, a structural technique that extracts features from the contours of characters, and a statistical approach. The combined system (tested on the same database) reported a correct classification rate of 93.05%, but had a zero substitution rate to achieve 100% reliability. Srihari [47], presents several different approaches ranging from stroke based recognizers to structural contour based chain code classifiers. The classifiers were based on different methodologies: statistical, structural, and syntactic. Each approach was trained on a set of 18,468 digits and tested on a set of 2,711 digits with correct classification results ranging from 83.6% to 96.4%. The data was derived from handwritten addresses obtained from the US Postal Service. Le Cun et al. [25] achieved good results with a back propagation neural network using size-normalized images as direct input. With a training and testing set size of 7291 and 2007 handwritten digits respectively, the system achieved a correct classification result of 92%.

CHAPTER 7
CONCLUSIONS

In this dissertation we have presented a new skeletonization algorithm derived from Voronoi diagrams. Our approach relies on a new method for constructing Voronoi diagrams of polygons that is robust, easy to implement and attractive for practical applications. We have shown that Voronoi skeletons are connected, accurate (characterized by Euclidean metrics), and are thus overcome some limitations of skeletons obtained by existing thinning algorithms. Redundant edges were deleted in a pruning step that was guaranteed to preserve connectivity. The technique implicitly provides a graph representation that enables the simple extraction of end points and junctions. Feature vectors consisting of Voronoi skeletons of character shapes were shown to be well suited for character recognition. Correct classification rates of 97.66%, 98.56% and 98.98% for the entire database, characters 'A'-'Z', and numeric characters '0'-'9' respectively, were obtained, when a neural network was trained with Voronoi skeleton representations of the character shapes. On an unconstrained NIST database, results comparable with the best results in the literature were obtained. A reliability rate of upto 98.08% with a substitution rate of 1.92% was observed for a database containing over 6000 numerals. These results suggest that Voronoi skeletons can provide efficient shape descriptors for character recognition.

Our future research efforts will be directed towards other aspects of character recognition. In particular we will study the character recognition problem for pen-based computers. Our skeletonization algorithm makes it easy to extract structural information such as end-points and junctions that are used in many syntactic recognition schemes. Since the skeleton is output in the form of a planar straight line graph, a single graph traversal is sufficient to determine end points and junctions. We shall investigate whether syntactic character recognition schemes based on Voronoi skeletons would provide efficient representations of more complex patterns.

REFERENCES

[1] C. Arcelli and G. S. Baja [1978], "On the sequential approach to medial line transformation," *IEEE Trans. Syst. Man Cybern.*, Vol. 8, No. 2, 139–144.

[2] C. Arcelli and G. S. Baja [1985], "A width–independent fast thinning algorithm," *IEEE Trans. Patt. Anal. Mach. Intell.*, Vol. 7, 463–474.

[3] F. Aurenhammer [1991], "Voronoi diagrams–a survey of a fundumental geometric data structure," *ACM Computing Surveys*, Vol. 23, 345–405.

[4] H. S. Baird [1993], "Recognition technology frontiers," *Pattern Recognition Letters* Vol 14, 327–334.

[5] H. Blum [1967], "A transformation for extracting new descriptors of shape," *Models for the Perception of Speech and Visual Form* (W. Wathen-Dunn, ed.), Cambridge MA, MIT Press.

[6] H. Blum and R. S. Nagel [1978], "Shape description using weighted symmetric axis features," *Pattern Recognition*, Vol. 10, 167–180.

[7] G. Borgefors and G. S. Baja [1988], "Skeletonizing the distance transform on the hexagonal grid," *Proc. Ninth Int. Conf. Patt. Recogn.* Rome, Italy, Vol. 1, 504–507.

[8] M. Brady and H. Asada [1984], "Smoothed local symmetries and their implementation," *International Journal of Robotics Research,* Vor. 3, No. 3, 36–61.

[9] J. W. Brandt and V. Ralph Algazi [1992], "Continuous skeleton computation by voronoi diagram," *Computer Vision, Graphics and Image Processing: Image Understanding*, Vol. 55, No. 3, 329–338.

[10] R. A. Brooks [1981], "Symbolic reasoning among 3-D models and 2-D images," *Artificial Intelligence*, Vol. 17, 285–348.

[11] R. G. Casey and K. Y. Wong [1990], "Document analysis systems and techniques," *Image Analysis Applications* (R. Kasturi and M. M. Trivedi, eds.), New York, Marcel Dekker.

[12] P. E. Danielsson [1980], "Euclidean distance mapping," *Comput. Graphics Image Processing*, Vol 14, 227-248.

[13] E. R. Davies and A. P. N. Plummer [1981] , "Thinning algorithms: a critique and a new methodology," *Pattern Recognition,* Vol. 14, 53–63.

[14] A. R. Dill, M. D. Levine, and P. B. Noble [1987], "Multiple resolution skeletons," *IEEE Trans. Patt. Anal. Mach. Intell.* Vol. 9, No. 4, 495–504.

[15] L. Dorst [1986], "Pseudo-Euclidean skeletons," *Proc. Eighth Int. Conf. Patt. Recogn.*, Paris, France, 286–288.

[16] S. Fortune [1987], "A sweepline algorithm for voronoi diagrams," *Algorithmica*, Vol. 2, 153–174.

[17] R. C. Gonzalez and R. E. Woods [1992], *Digital Image Processing*, New York, Addison-Wesley.

[18] C. J. Hilditch [1969], "Linear skeletons from square cupboards," *Machine Intell.*, Vol. 4, 403-420.

[19] S. Kahan, T. Pavlidis, and H. S. Baird [1987], "On the recognition of printed characters of any font and size," *IEEE Trans. Patt. Anal. Mach. Intell.,* Vol. 9, No. 2, 274–288.

[20] B. Kalman [1990], "Super linear learning in back propagation neural nets," Dept. of Computer Science Technical Report WUCS-90-21, Washington University, St. Louis.

[21] A. Laine, W. E. Ball and Arun Kumar [1991], "A multiscale approach for recognizing complex annotations in engineering drawings," *Proc. Conf on Computer Vision and Pattern Recognition*, Lahaina, Hawaii.

[22] A. Laine, S. Schuler, and V. Girish [1993], "Wavelet representations for recognizing complex annotations," *Machine Vision and Applications*, Vol. 6, 110–123.

[23] L. Lam, S. W. Lee and C. Y. Suen [1988], "Structural classification and relaxation matching of totally unconstrained handwritten zip-code numbers," *Pattern Recognition*, Vol. 21, No. 1, 19–31.

[24] L. Lam, S. W. Lee and C. Y. Suen [1992], "Thinning methodologies–a comprehensive survey," *IEEE Trans. Patt. Anal. Mach. Intell.*, Vol. 14, No. 9, 885.

[25] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, and H. S. Baird [1990], "Constrained neural network for unconstrained handwritten digit recognition," *Proc. International Workshop on Frontiers in Handwriting Recognition*, Concordia University, Montreal, 145-154.

[26] D. T. Lee [1982], "Medial axis transformation of a planar shape," *IEEE Trans. Patt. Anal. Mach. Intell.* Vol. 4, No. 4, 363–369.

[27] D. T. Lee and L. Drysdale [1981], "Generalization of voronoi diagrams in the plane," *SIAM J. Computing*, Vol. 10, No. 1, 73–87.

[28] F. Leymarie and M. D. Levine [1992], "Simulating the grass-fire transform using an active contour model," *IEEE Trans. Patt. Anal. Mach. Intell.* Vol. 14, No. 1, 56–75.

[29] M. Leyton [1987], "Symmetry–Curvature duality," *Computer Vision, Graphics, and Image Processing*, Vol. 38, 327–341.

[30] D. C. Marr and E. Hildreth [1980], "Theory of edge detection," *Proc. Royal Soc. London.,* B 207, 187–217.

[31] S. N. Meshkat and M. Sakkas [1987], "Voronoi diagram for multiply connected polygonal domains II: implementation and application," *IBM J. Res. Develop.* Vol. 31, No. 3, 373–381.

[32] S. Mori, C. Y. Suen and K. Yamamoto [1992], "Historical review of OCR research and development," *Proc. IEEE*, Vol. 80, No. 7. 1029–1058.

[33] U. Montanari [1968], "A method for obtaining skeletons using a quasi–Euclidean distance," *J. Assoc. Comput. Machinery*, Vol. 15, 60–624.

[34] U. Montanari [1969], "Continuous skeletons from digitized images," *J. Assoc. Comput. Machinery*, Vol. 16, No. 4, 534-549.

[35] C. Nadal and C. Y. Suen [1988], "Recognition of totally unconstrained handwritten digit by decomposition and vectorisation," Technical Report, Concordia University, Montreal.

[36] G. Nagy [1992], "At the frontiers of OCR," *Proceedings of the IEEE*, Vol. 80, No. 7, 1093–1100.

[37] R. Ogniewicz [1994], "Skeleton-space: a multiscale shape description combining region and boundary information," *Proc. Conf. on Computer Vision and Pattern Recognition*, Seattle, Washington, 746–751.

[38] R. Ogniewicz and M. Ilg [1992], "Voronoi skeletons: theory and applications," *Proc. Conf. on Computer Vision and Pattern Recognition*, Champaign, Illinois, 63–69.

[39] Theo Pavlidis [1982], *Algorithms for Graphics and Image Processing*, Maryland, Computer Science Press.

[40] T. Pavlidis [1982], "An asynchronous thinning algorithm," *Computer Vision, Graphics and Image Processing*, Vol. 20, 133–157.

[41] T. Pavlidis [1993], "Recognition of printed text under realistic conditions," *Pattern Recognition Letters,* Vol. 14, 317–326.

[42] S. M. Pizer, W. R. Oliver and S. H. Bloomberg [1987], "Hierarchical shape description via the multiresolution symmetric axis transform," *IEEE Trans. Patt. Anal. Mach. Intell.*, Vol. 9, No. 4, 505–511.

[43] F. P. Preparata and M. I. Shamos [1985], *Computational Geometry–An Introduction*, New York, Springer Verlag.

[44] H. Rom and G. Medioni [1993], "Hierarchical decomposition and axial shape description," *IEEE Patt. Anal. Mach. Intell.*, Vol. 15, No. 10, 973–981.

[45] A. Rosenfeld and J. L. Pfaltz [1966], "Sequential operations in digital picture processing," *J. Assoc. Comput. Machinery*, Vol. 13, 471–494.

[46] L. Schomaker [1993], "Using stroke or character based self–organizing maps in the recognition of online, connected cursive script," *Pattern Recognition*, Vol. 26, No. 3, 443–450.

[47] S. N. Srihari [1993], "Recognition of handwritten and machine-printed text for postal address interpretation," *Pattern Recognition Letters*, Vol. 14, 291–302.

[48] V. Srinivasan & L. R. Nackman [1987], "Voronoi diagram for multiply connected polygonal domains I: Algorithm," *IBM J. Res. Develop.*, Vol. 31, No. 3, 361–372.

[49] V. Srinivasan, L. R. Nackman, J. Tang and S.N. Meshkat [1992], "Automatic mesh generation using the symmetric axis transformation of polygonal domains," *Proc. IEEE*, Vol. 80, No. 9, 1485–1501.

[50] J. Subrahmonia, D. Keren and D. B. Cooper [1993], "Recognizing mice, vegetables and handprinted characters based on implicit polynomials, invariants and bayesian methods," *Proc. Fourth International Conference on Computer Vision*, Berlin, Germany, 320–324.

[51] C. Y. Suen, R. Legault, C. Nadal, M. Cheriet and L. Lam [1992], "Computer recognition of unconstrained handwritten numerals," *Proc. IEEE* Vol. 80, No. 7, 1162–1180.

[52] C. Y. Suen, R. Legault, C. Nadal, M. Cheriet and L. Lam [1993], "Building a new generation of handwriting recognition systems," *Pattern Recognition Letters* Vol. 14, 303–315.

[53] C. K. Yap [1984], "An $O(n \log n)$ algorithm for the voronoi diagram of a set of simple curve segments," NYU-Courant Robotics Report No. 43, New York University, New York.

BIOGRAPHICAL SKETCH

Niranjan Mayya was born on June 18, 1966, in Pune, India. He received his undergraduate degree in computer engineering from the University of Poona, India, in July, 1987. He received his Master of Science degree in computer and information sciences at the University of Florida, Gainesville, in June, 1991. He will receive his Ph.D in the same major in August, 1994. His research interests include character recognition, pattern recognition and geometric algorithms for computer vision.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Andrew F. Laine, Chairman
Assistant Professor of Computer and Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Baba C. Vemuri, Cochairman
Associate Professor of Computer and Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Gerhard X. Ritter
Professor of Computer and Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Sartaj Sahni
Professor of Computer and Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Murali Rao
Professor of Mathematics

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Richard A. Kiltie
Associate Professor of Zoology

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August 1994

[illegible]

Winfred M. Phillips
Dean, College of Engineering

Karen A. Holbrook
Dean, Graduate School